ДИСКРЕТНАЯ МАТЕМАТИКА

ФН, магистры - 2 семестр

Лекция 2. ТЕОРИЯ ГРАФОВ

Первой работой теории графов как математической дисциплины считают статью Эйлера (1736 г.), в которой рассматривалась задача о Кёнингсбергских мостах. Эйлер показал, что нельзя обойти семь городских мостов и вернуться в исходную точку, пройдя по каждому мосту ровно один раз.
Следующий импульс теория графов получила спустя почти 100 лет с развитием исследований по электрическим сетям, кристаллографии, органической химии и другим наукам.

2.1. Основные определения

Графы есть способ „визуализации“ связей между определенными объектами. Связи эти могут быть „направленными“, как, например, в генеалогическом древе, или “ненаправленными“ (сеть дорог с двусторонним движением). В соответствии с этим в теории графов выделяют два основных типа графов: ориентированные (или направленные) и неориентированные.
Построение математического определения графа осуществляется путем формализации и „объектов“, и „связей“ как элементов некоторых (как правило, конечных) множеств.

Неориентированные графы

Неориентированный граф G задается двумя множествами

$$G = (V, E),$$

где V — конечное множество, элементы которого называют **вершинами**, E — множество неупорядоченных пар на V, элементы которого называют **ребрами**.

$u \mapsto v$ —ребро соединяет вершины u и v, $\{u, v\} \in E$, .

Вершины u и v, для которых $u \mapsto v$, называют **смежными**, а также **концами ребра** $\{u, v\}$.

Если $u \mapsto v$, говорят, что вершины u и v связаны **отношением непосредственной достижимости**.

Ребро e называют **инцидентным** вершине v, если она является одним из его концов.

$\Gamma(v) = \{x\colon x \mapsto v\}$ — множество вершин смежных с вершиной v .

Степенью вершины v называется число $\mathrm{dg}(v)$ всех инцидентных ей ребер.

$\mathrm{dg}(v) = |\Gamma(v)|$.

Лемма „о рукопожатиях“

Сумма степеней всех вершин графа равна удвоенному числу ребер.

Цепь в неориентированном графе G — это последовательность вершин (конечная или бесконечная) $v_0, v_1, \ldots, v_n, \ldots$, такая, что для любого i $v_i \longmapsto v_{i+1}$, если v_{i+1} существует.

Вершина v неориентированного графа G **достижима** из вершины u этого графа ($u \,|{=}{=}|^* v$), если существует цепь $v_0, v_1, \ldots, v_n$ такая, что $u = v_0$, $v_n = v$. Вершины u и v, называют **концами цепи**.

Отношение достижимости $\longmapsto$ в неориентированном графе задано как **рефлексивно-транзитивное замыкание** отношения $\to$ непосредственной достижимости.

Отношение достижимости в неориентированном графе $u \,|{=}{=}|^* v$ рефлексивно, симметрично и транзитивно, отношение достижимости в неориентированном графе есть отношение эквивалентности.

Простая цепь — это цепь, все вершины которой, кроме, быть может, первой и последней, попарно различны, и все ребра попарно различны.

Простую цепь ненулевой длины с совпадающими концами называют **циклом**.

Граф, не содержащий циклов, называют **ациклическим**

Ориентированные графы

Ориентированный граф (орграф) G задается двумя множествами

$$G = (V, E),$$

где V — конечное множество, элементы которого называют **вершинами**, E — множество *упорядоченных пар* на V, т.е. подмножество множества $V \times V$, элементы которого называют **дугами**.

$u \to v$ —**дуга ведет** из вершины u в вершину v.

Вершины u и v, для которых $u \to v$, называют **смежными**,

u — **начало**, а v — **конец дуги** (u, v).

Дугу, начало и конец которой есть одна и та же вершина, называют **петлей**.

Если $u \to v$, то из вершины u непосредственно достижима вершина v.

Вершины u и v связаны **отношением непосредственной достижимости**.

Дугу (u, v) называют **заходящей** в вершину v и **исходящей** из вершины u. Дугу называют **инцидентной** вершине v, если она или заходит в v или исходит из v.

Полустепенью захода вершины v называют число $\mathrm{dg}^-(v)$ заходящих в нее дуг, а **полустепенью исхода** вершины v — число $\mathrm{dg}^+(v)$ исходящих из нее дуг.

Степень вершины v, обозначаемая $\mathrm{dg}(v)$ — это сумма полустепеней захода и исхода.

Множество $\Gamma(v) = \{x\colon v \to x\}$ называют множеством **преемников** вершины v, множество $\Gamma^{-1}(v) = \{x\colon x \to v\}$ — множеством **предшественников** вершины v.

$$\mathrm{dg}^+(v) = |\Gamma(v)|, \mathrm{dg}^-(v) = |\Gamma^{-1}(v)|.$$

Путь в ориентированном графе G — это последовательность вершин (конечная или бесконечная) $v_0, v_1, \ldots, v_n, \ldots$, такая, что для любого i $v_i \to v_{i+1}$, если v_{i+1} существует.

Для конечного пути $v_0, v_1, \ldots, v_n$ число n называют **длиной пути** $(n \geq 0)$.
Длина пути есть число его дуг, т.е. всех дуг, которые ведут из вершины v_i в вершину v_{i+1} ($i = 0, \cdots, n-1$).
Путь длины 0 — это произвольная вершина графа.

Простой путь — это путь, все вершины которого, кроме, быть может, первой и последней, попарно различны.

Простой путь ненулевой длины, начало и конец которого совпадают, называют **контуром**.

Если существует **путь** ненулевой длины, ведущий из u в v, то пишут $u \Rightarrow^+ v$.

Если необходимо явно указать длину пути, то пишут $u \Rightarrow^n v$ и говорят, что существует путь длины n ,ведущий из u в v.

Простой путь — это путь, все вершины которого, кроме, быть может, первой и последней, попарно различны.

Простой путь ненулевой длины, начало и конец которого совпадают, называют **контуром**.

Произвольный путь ненулевой длины, начало и конец которого совпадают, будем называть **замкнутым путем**.

Ориентированный граф, не содержащий контуров, называют **бесконтурным графом**.

Вершина v ориентированного графа G **достижима** из вершины u этого графа ($u \Rightarrow^* v$,) если существует путь $v_0, v_1, \ldots, v_n$, такой, что $u = v_0$, $v = v_n$;
вершина u —**начало**, вершина v — **конец** данного **пути**).

Отношение достижимости $\Rightarrow^*$ в ориентированном графе задано как **рефлексивно-транзитивное замыкание** отношения $\rightarrow$ непосредственной достижимости.

Отношение достижимости в ориентированном графе рефлексивно и транзитивно, в общем случае не **антисимметрично**: если две вершины ориентированного графа достижимы одна из другой, то из этого вовсе не следует, что они совпадают. Таким образом, отношение достижимости в ориентированном графе есть *отношение предпорядка*.

Подграф.

Определение 2.1. Неориентированный (ориентированный) граф $G_1 = (V_1, E_1)$ называют **подграфом** неориентированного (ориентированного) графа $G = (V, E)$, если $V_1 \subseteq V$ и $E_1 \subseteq E$.

Обозначение $G_1 \subseteq G$.

Подграф G_1 называют **собственным подграфом** графа G,Если хотя бы одно из указанных двух включений в определении 2.1 строгое.

Подграф G_1 называют **остовным подграфом** графа G, если $V_1 = V$.

Подграф G_1 неориентированного (ориентированного) графа G называют **подграфом, порожденным множеством вершин** $V_1 \subseteq V$, если каждое ребро (дуга) тогда и только тогда принадлежит $E_1 \subseteq E$, когда его (ее) концы принадлежат V_1. Часто в случае, если множество вершин V_1 подразумевается, говорят просто о **порожденном подграфе**.

Подграф графа G, порожденный множеством вершин V_1, в отличие от произвольного подграфа графа G с множеством вершин V_1, должен включать все ребра (дуги), концы которых принадлежат множеству V_1.

Подграф $G_1 \subseteq G$ называют **максимальным подграфом**, обладающим данным свойством P, если он не является собственным подграфом никакого другого подграфа графа G, обладающего свойством P.

Определение 2.2. Неориентированный граф называют **связным**, если любые две его вершины u и v соединены цепью ($u \,|{=}{=}|^{*}\, v$).

Компонента связности (или просто **компонента**) неориентированного графа — это его максимальный связный подграф.
В неориентированном графе две вершины, соединенные цепью, связаны отношением достижимости, которое является эквивалентностью. Компонента графа — это подграф, порожденный некоторым классом эквивалентности вершин по отношению достижимости.
Две различные компоненты не пересекаются.

Определение 2.3. Для ориентированного графа определяют понятия **связности**, **сильной связности** и **слабой связности**.
Ориентированный граф называют **связным**, если для любых двух его вершин u, v вершина v достижима из вершины u или вершина u достижима из вершины v ($u \Rightarrow^{*} v$ или $v \Rightarrow^{*} u$).

Компонента связности (или просто **компонента**) ориентированного графа — это максимальный связный подграф.
Компоненты ориентированного графа могут пересекаться (отношение достижимости в ориентированном графе не является эквивалентностью).

Определение 2.4. Ориентированный граф называют **сильно связным**, если для любых двух его вершин u и v вершина v достижима из вершины u и вершина u достижима из вершины v $(u \Rightarrow^* v$ и $v \Rightarrow^* u)$.
Бикомпонента ориентированного графа — это его максимальный сильно связный подграф.

Если $u \Rightarrow^* v$ и $v \Rightarrow^* u$, то u и v связаны **отношением взаимной достижимости**.
Это бинарное отношение рефлексивно, симметрично и транзитивно, т.е. является отношением эквивалентности. Следовательно, две различные бикомпоненты не пересекаются, т.е. не имеют ни общих вершин, ни общих ребер.

Примеры.

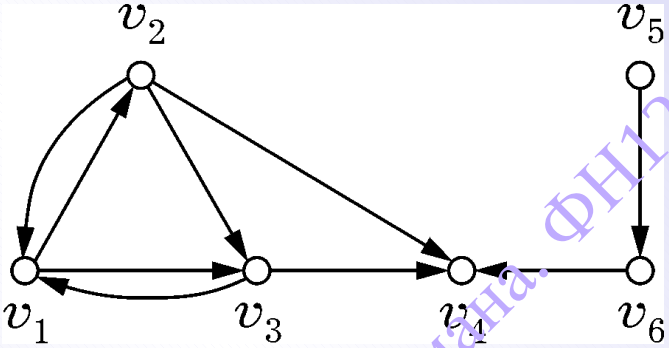

Рис. 1

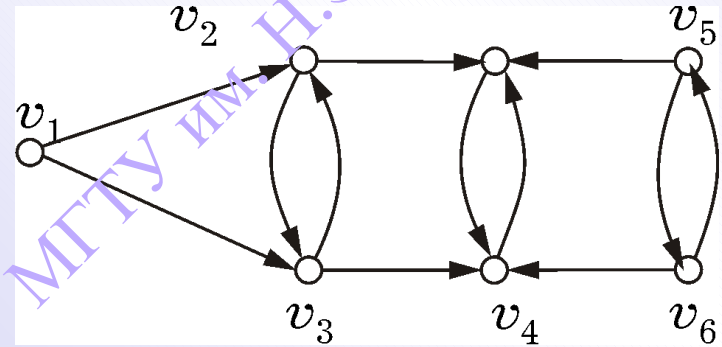

Рис. 2

Определение 2.5. Неориентированный граф $G_1 = (V_1, E_1)$ называют **ассоциированным** с ориентированном графом $G = (V, E)$, если его множество вершин совпадает с множеством вершин ориентированного графа G, а пара $\{u, v\}$ образует ребро тогда и только тогда, когда $u \neq v$ и из u в v или из v в u ведет дуга, т.е. $V_1 = V$ и

$$E_1 = \{\{u, v\} \colon (u, v) \in E \text{ или } (v, u) \in E,\ u \neq v\}.$$

Определение 2.6. Ориентированный граф называют **слабо связным**, если ассоциированный с ним неориентированный граф связный. **Компонентой слабой связности (слабой компонентой)** ориентированного графа называют его максимальный слабо связный подграф.

2.2. Способы представления

Три наиболее распространенных способа представления графов (кроме рисунков и пары множеств).

Пронумеруем начиная с единицы все **вершины** и все **ребра неориентированного графа** (**дуги**, включая *петли* **ориентированного графа**).

Граф (неориентированный или ориентированный) может быть представлен в виде матрицы типа $n \times m$, где n — число вершин, а m — число ребер (или дуг).

Для неориентированного графа:

$$\begin{cases} a_{ij} = 1, \text{ для } i\text{-й вершины } j\text{-е } \textit{ребро инцидентное}; \\ 0, \text{ иначе.} \end{cases}$$

Для ориентированного графа:

$$\begin{cases} a_{ij} = 1, \text{ для } i\text{-й вершины } j\text{-я } \textit{дуга выходящая}; \\ -1, \text{ для } i\text{-й вершины } j\text{-я дуга } \textit{заходящая}; \\ 0, \text{ иначе.} \end{cases}$$

Матрицу (a_{ij}) типа $n \times m$, определенную указанным образом, называют **матрицей инциденций**.

Пример 2.1. Ориентированный граф $G = (V, E)$ задан множеством вершин и множеством упорядоченных пар

$$V = \{v_1, v_2, v_3\}$$
$$E = \{(v_1, v_2), (v_1, v_3), (v_2, v_3), (v_3, v_2)\}.$$

Нумерация вершин v_i уже задана индексом i.
Зададим нумерацию дуг: дуге (v_1, v_2) присвоим номер 1, дуге (v_1, v_3) — 2, дуге (v_2, v_3) — 3 и дуге (v_3, v_2) — 4.
Матрицу инциденций удобно заполнять по столбцам, записывая для k-й дуги (v_i, v_j) 1 в i-й и -1 в j-й строках k-го столбца и 0 во всех остальных строках k-го столбца. В результате получим матрицу

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ -1 & 0 & 1 & -1 \\ 0 & -1 & -1 & 1 \end{pmatrix}$$

Практически способ представления графа в виде матрицы инциденций неэффективен.

Задача. С помощью матрицы инциденций в ориентированном графе для данной вершины v_k найти ее „окружение“ — множество всех вершин, непосредственно достижимых из v_k — **преемников** и множество всех вершин, из которых она непосредственно достижима — **предшественников** вершины v_k. (Пусть $k = 3$)

$$\begin{pmatrix} 1 & 1 & 0 & 0 \\ -1 & 0 & 1 & -1 \\ 0 & -1 & -1 & 1 \end{pmatrix}$$

На матрице инциденций ориентированного графа нужно идти по строке с номером k до появления ненулевого элемента ($+1$ или -1).
При обнаружении -1, в соответствующем столбце надо найти строку, в которой записана 1, и получить номер вершины v_i, из которой непосредственно достижима v_k. ($i = 1, j = 2$)
В случае обнаружения $+1$, в соответствующем столбце надо найти строку, в которой записано число -1. Номер строки, в которой стоит -1, дает номер вершины v_n, непосредственно достижимой из v_k .($n = 2$)
Для получения всего „окружения“ надо проделать указанный поиск для всех ненулевых элементов k -й строки.

Временные затраты на решение задачи.

Наиболее трудоемкая процедура — поиск ненулевого элемента в столбце. Число таких процедур поиска равно степени вершины v_k .

Сложность алгоритма анализа окружения вершины v_k составляет $O(\deg v_k)$.

Поиск „окружения“ всех вершин займет время порядка $|V| \sum \deg v_k \sim |E|$.

Представление графа матрицей смежности вершин.
Матрица смежности вершин, или **булева матрица** графа — это квадратная матрица B порядка n, элементы которой определяют следующим образом: для неориентированного графа

$$\begin{cases} b_{ij} = 1, i\text{-я и } j\text{-я вершины } \textit{смежные}; \\ 0, \text{иначе}; \end{cases}$$

для ориентированного графа

$$\begin{cases} b_{ij} = 1, \text{из } i\text{-й вершины в } j\text{-ю ведет дуга}; \\ 0, \text{иначе}; \end{cases}$$

В k-й строке матрицы ориентированного графа количество единиц равно **полустепени исхода** $\deg^+ v_k$ вершины v_k, а количество единиц в k-м столбце — **полустепени захода** $\deg^- v_k$.
Для неориентированного графа **матрица** смежности вершин **симметрическая**.

Пример 2.2. Для ориентированного графа $G = (V,\ E)$, где

$$V = \{v_1,\ v_2,\ v_3\}\ E = \{\{v_1, v_2\},\ \{v_1, v_3\},\ \{v_2, v_3\},\ \{v_3, v_2\}\}.$$

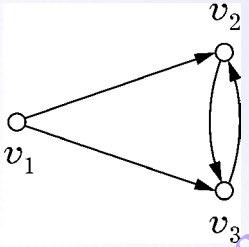

Рис. 3

матрица смежности вершин имеет вид

$$\begin{pmatrix} 0 & 1 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$$

Задача: поиск „окружения“ с использованием матрицы смежности вершин. Для определения „окружения“ вершины v_k нужно сначала идти по k -й строке матрицы и искать ненулевые элементы.
Если элемент $a_{ki} = 1$, то вершина v_i достижима из вершины v_k .
После просмотра k -й строки надо просмотреть k -й столбец. Если элемент $a_{jk} = 1$, то вершина v_k достижима из вершины v_j .

Динамическое создание графа. Списки смежности

Рассмотрим ориентированный граф. Для задания множества **вершин**, **непосредственно достижимых** из вершины v, используют **линейный однонаправленный список**.

Каждый элемент списка смежности вершины v включает данные (номер вершины u, для которой есть дуга $v \to u$) и указатель на следующий элемент списка (очередную вершину, достижимую из v). Список в целом задается указателем на его первый элемент (**голову списка**). Последний элемент списка содержит „пустой“ указатель.

Список смежности вершины может при необходимости дополняться. Для этого в последнем элементе списка „пустой“ указатель заменяется указателем на добавляемый элемент, который становится последним элементом списка с „пустым“ указателем.

Если количество вершин ориентированного графа известно заранее, то ориентированный граф удобно задавать в виде **массива лидеров**.

Массив лидеров — матрица-столбец, элементами которой являются „головы“ (первые элементы) списков смежности вершин ориентированного графа. Число элементов массива лидеров равно числу вершин графа.

Рис. 4

Пример 2.3. Ориентированный граф $G = (V, E)$ задан множеством вершин и множеством упорядоченных пар

$$V = \{v_1, v_2, v_3, v_4, v_5\}$$
$$E = \{(v_1, v_2), (v_2, v_2), (v_2, v_3), (v_2, v_4), (v_2, v_5), (v_3, v_1), (v_5, v_2)\}$$

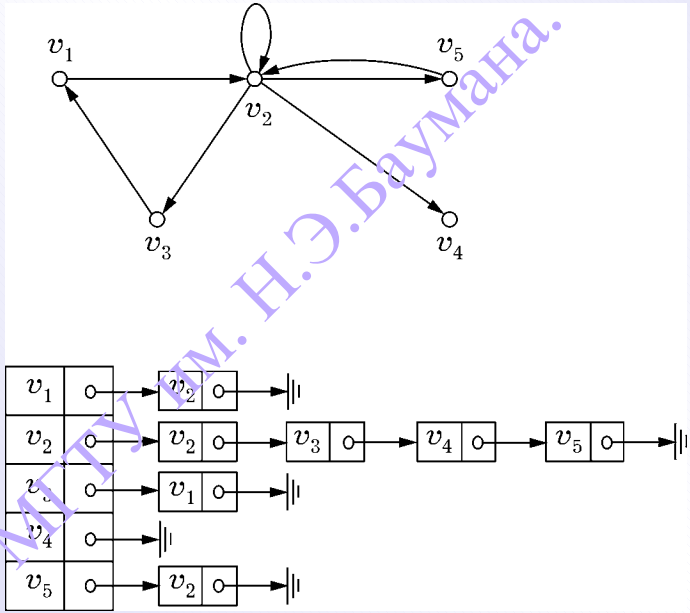

Рис. 5

С использованием списков смежности просто решается задача поиска **преемников** данной вершины: для этого достаточно просмотреть список смежности вершины.

Менее эффективно решается задача поиска **предшественников** вершины, так как в этом случае необходимо, просмотреть списки смежности всех вершин с целью поиска в них данной вершины.

Неориентированный граф задать с помощью списков смежности можно так же, как и ориентированный. Здесь в список смежности вершины v войдут все вершины, смежные с ней, а списки смежности могут быть собраны в массив лидеров. Для неориентированого графа задача поиска „окружения“ одной вершины требует однократного просмотра ее списка смежности.

2.3. Деревья

Определение 2.7. **Неориентированным деревом** называют **связный** и **ациклический неориентированный граф**.

Определение 2.8. **Ориентированным деревом** называют **бесконтурный ориентированный граф**, у которого **полустепень захода** любой вершины не больше 1 и существует ровно одна вершина, называемая **корнем ориентированного дерева**, **полустепень захода** которой равна 0.

В ориентированном дереве любая вершина **достижима** из корня.
Требование бесконтурности ориентированного графа в определении 2.8 является обязательным.

Определение 2.9. Вершину v ориентированного дерева называют **потомком (подлинным потомком)** вершины u, если существует путь из u в v (путь ненулевой длины из u в v). В этом же случае вершину u называют **предком (подлинным предком)** вершины v, а если длина пути из u в v равна 1, то вершину v называют **сыном** вершины u, которая при этом вполне естественно именуется **отцом** вершины v.
Вершину, не имеющую потомков, называют **листом**.

Пример

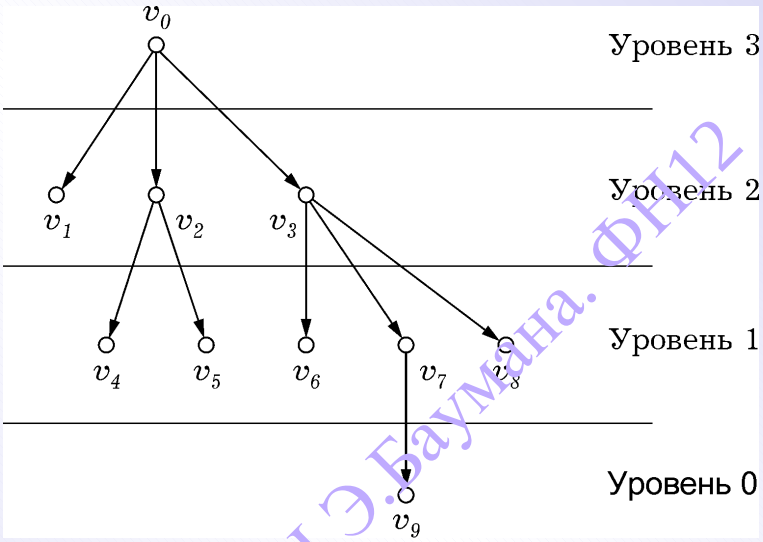

Рис. 6

Вершины v_4 и v_5 — сыновья вершины v_2, которая, в свою очередь, является сыном вершины v_0 — корня дерева.

Вершины v_4 и v_5 являются подлинными потомками вершин v_0. и v_2, которые соответственно будут их подлинными предками. Вершины v_1, v_4, v_5, v_6, v_9, v_8 — листья дерева.

Взаимно недостижимые вершины ориентированного дерева (v_2 и v_9) не являются ни предком, ни потомком одна другой. Каждая вершина будет сама для себя предком и потомком, но не подлинным.

Определение 2.10. Ориентированное дерево, у которого каждая вершина, отличная от корня, есть лист, называют **кустом**.

Определение 2.11. Подграф неориентированного (ориентированного) дерева, являющийся неориентированным (ориентированным) деревом, называют **поддеревом** исходного дерева.

Компонентами ориентированного дерева являются его **подграфы**, порожденные множеством вершин, расположенных на некотором пути из корня в лист.

Подграф, порожденный множеством вершин $\{v_3,\ v_6,\ v_7,\ v_8,\ v_9\}$, является **поддеревом** ориентированного дерева.

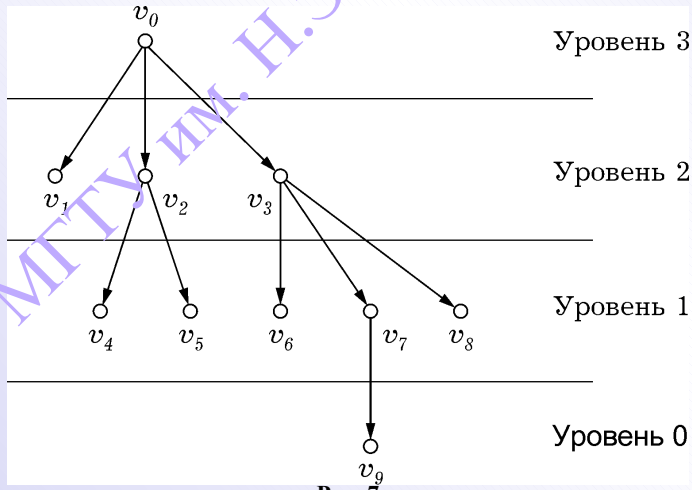

Рис. 7

Определение 2.12. Произвольный ациклический граф называют **неориентированным лесом**.

Если каждая **слабая компонента** ориентированного графа является ориентированным деревом, то такой граф называют **ориентированным лесом**.

Неориентированный лес — это неориентированный граф, каждая компонента которого является неориентированным деревом.

Примеры неориентированного и ориентированного леса

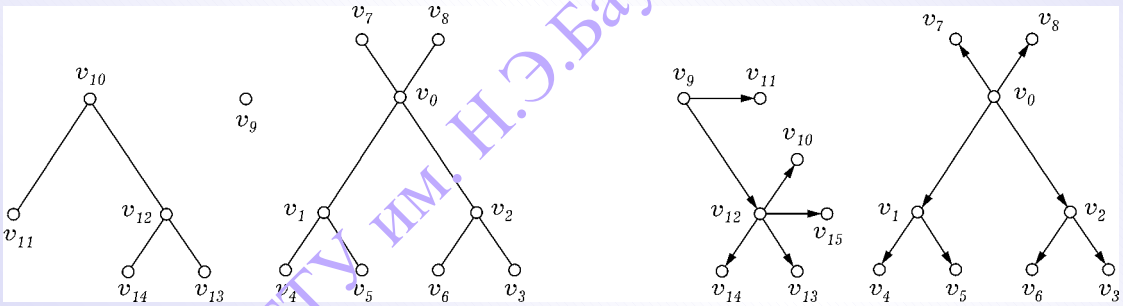

Рис. 8

Остовным лесом (**деревом**) неориентированного (ориентированного) графа называют любой его остовный подграф, являющийся лесом (деревом).

Определение 2.13. **Высота ориентированного дерева** — это наибольшая длина пути из корня в лист.
Глубина $\mathrm{d}(v)$ **вершины ориентированного дерева** v — это длина пути из корня в эту вершину.
Высота $\mathrm{h}(v)$ **вершины ориентированного дерева** v — это наибольшая длина пути из данной вершины в лист.
Уровень вершины ориентированного дерева — это разность между высотой ориентированного дерева и глубиной данной вершины.

Уровень корня равен высоте ориентированного дерева, но уровни различных листьев,так же как и их глубины, могут быть различными; **высота** любого **листа** равна **нулю**.

Определение 2.14. Ориентированное дерево называют **бинарным**, если полустепень исхода любой его вершины не больше 2 (БОД).
Бинарное ориентированное дерево называют **полным**, если из любой его вершины, не являющейся листом, исходят ровно две дуги, а уровни всех листьев совпадают.

Теорема 1 (теорема о высоте бинарного ориентированного дерева с заданным числом листьев). Бинарное ориентированное дерево с n листьями имеет высоту, не меньшую $\log_2 n$.

◄ В **полном** бинарном ориентированном дереве (ПБОД) высоты h ровно 2^h листьев. Используем метод математической индукции по высоте ориентированного дерева.

Ориентированное дерево высоты 0 имеет $2^0 = 1$ лист. ПБ ОД высоты 1 имеет $2^1 = 2$ листа.

Пусть ПБОД имеет высоту k и 2^k листьев.

Рассмотрим ПБОД высоты $k+1$. Поскольку в ПБОД уровни всех листьев совпадают, ПБОД высоты $k+1$ можно получить из ПБОД высоты k, если из каждого листа последнего провести по две дуги.

Количество листьев в ПБОД высоты $k+1$ будет в 2 раза больше, чем в ПБОД высоты k, т.е. $2^k \cdot 2 = 2^{k+1}$.

В произвольном БОД листьев может быть только меньше, чем в полном.

В произвольном БОД высоты h не более 2^h листьев $(n \leq 2^h)$.

$h \geq \log_2 n$ ►

Задача сортировки

Пусть, что необходимо расположить строго по возрастанию элементы конечного линейно упорядоченного множества $\{a_1, \ldots, a_n\}$.

Алгоритм сортировки

Все сравнения, которые могут быть проведены в процессе работы некоторого алгоритма, изображаются в виде ориентированного дерева, называемого **деревом решений**.

Пример 2.4. Задача сортировки трехэлементного множества $\{a, b, c\}$.

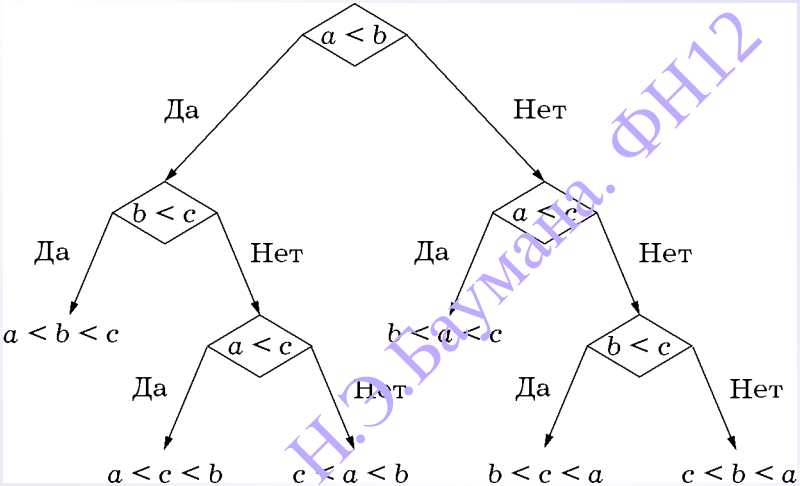

Рис. 9

Дерево решений для трехэлементного множества $\{a, b, c\}$.

Вершины ориентированного дерева,не являющиеся листьями, помечены проверяемыми неравенствами, а множество листьев находится во взаимно однозначном соответствии с множеством всех перестановок исходного множества.

Алгоритм сортировки должен найти такую **перестановку** $\{a_{p_1}, \ldots, a_{p_n}\}$ элементов множества, которая была бы согласована с заданным на нем отношением $\leq$ линейного порядка, т.е. для любых k, l из справедливости неравенства $p_k < p_l$ должно следовать $a_{p_k} \leq a_{p_l}$.

Первоначально элементы могут быть расположены в произвольном порядке, т.е. исходной может быть любая перестановка элементов сортируемого множества, никакой априорной информации об этой перестановке не имеем.

Для получения такой информации надо попарно сравнивать элементы a_i в какой-либо последовательности.

Проводить все возможные попарные сравнения не обязательно. Можно использовать транзитивность отношения порядка, т.е. $a_{p_k} \leq a_{p_l}$ и $a_{p_l} \leq a_{p_m}$, то $a_{p_k} \leq a_{p_m}$.

В результате сортировки может получиться любая перестановка исходного множества, каждой такой перестановке соответствует лист дерева решений, в общем случае количество листьев будет равно $n!$ — количеству перестановок n -элементного множества.

Следовательно, сортируя входную последовательность, алгоритм обязательно пройдет какой-то путь от корня дерева решений к одному из листьев, и, таким образом, число операций сравнения (число шагов алгоритма сортировки) будет величиной, пропорциональной высоте дерева решений, не меньшей чем $\log_2 n!$, в силу теоремы 1.

Используя для оценки факториала при больших n **формулу Стирлинга** $n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)^n$, получаем, что дерево решений имеет высоту порядка $n \log_2 n$.

материал для самостоятельного изучения

2.4. Остовное дерево наименьшего веса

Неориентированный (ориентированный) граф, у которого каждому ребру (дуге) сопоставлено некоторое действительное число, называют **взвешенным** или **размеченным** графом.
Это число называют **весом** или **меткой** ребра (дуги).
Алгоритм Краскала вычисляет для заданного взвешенного неориентированного графа G *остовное дерево* с наименьшей суммой весов ребер — **остовное дерево наименьшего веса**.

При описании алгоритма будем использовать способ хранения данных, называемый **очередью**.
Элементы данных в очереди упорядочиваются по времени поступления. Элементы можно добавлять в очередь и извлекать из очереди.
В каждый момент времени доступен только один элемент, который был помещен в очередь раньше других, — „голова“ очереди.
При добавлении новый элемент помещается в „хвост“ очереди, т.е. работа ведется по обычному для очереди правилу — „первым пришел — первым вышел“.
Чтобы извлечь из очереди некоторый элемент, не доступный в текущий момент, надо извлечь все ранее поступившие элементы, начиная с „головы“ очереди.
Обычно очередь реализуется в виде списка.

Алгоритм Краскала

Рассмотрим алгоритм нахождения остовного дерева наименьшего веса. Пусть дан связный неориентированный граф $G = (V, E)$ с числовыми неотрицательными весами ребер. Вес ребра e обозначим $\varphi(e)$.

В результате работы алгоритма получим остовное дерево $T = (V, H)$ графа G, такое, что сумма $\sum_{e \in H} \varphi(e)$ является наименьшей.

Отсортируем все ребра исходного графа по возрастанию весов и сформируем из них очередь так, чтобы в „голове“ очереди находилось ребро с наименьшим весом, а в „хвосте“ — с наибольшим и веса ребер не убывали от „головы“ очереди к „хвосту“.

Метод состоит в „сшивании“ искомого дерева из *компонент* остовного леса. Первоначально *остовный лес* представляет собой множество изолированных вершин исходного графа, т.е. его множество ребер пусто. На первом шаге из очереди извлекается ребро наименьшего веса и добавляется к множеству ребер исходного дерева.

На последующих шагах алгоритма из очереди извлекается по одному ребру. Если это ребро соединяет вершины, принадлежащие разным компонентам текущего остовного леса, то оно добавляется к текущему множеству ребер искомого дерева, а указанные компоненты сливаются в одну. Иначе ребро отбрасывается. Процесс повторяется до тех пор, пока число компонент остовного леса не окажется равным 1. Можно показать, что эта компонента и будет искомым остовным деревом наименьшего веса.

1. Множество ребер H искомого остовного дерева полагаем пустым ($H = \varnothing$).

2. Формируем множество $V_S = \{\{v_1\}, \ldots, \{v_n\}\}$, элементами которого являются множества вершин, соответствующих компонентам исходного остовного леса. Каждая такая компонента состоит из единственной вершины.

3. Сортируем множество ребер E исходного графа по возрастанию весов и формируем очередь Q , элементами которой являются ребра графа G .

4. Если множество V_S содержит более одного элемента (т.е. остовный лес состоит из нескольких компонент) и очередь Q не пуста, переходим на шаг **5**, если иначе — на шаг **7**.

5. Извлекаем из очереди Q ребро e . Если *концы* ребра e принадлежат различным множествам вершин V_i и V_j из V_S , то переходим на шаг **6**, если иначе, то отбрасываем извлеченное ребро и возвращаемся на шаг **4**.

6. Объединяем множества вершин V_i и V_j (полагая $W = V_i \cup V_j$), удаляем множества V_i и V_j из множества V_S и добавляем в V_S множество W . Добавляем ребро e в множество H . Возвращаемся на шаг **4**.

7. Прекращаем работу. Множество H есть множество ребер полученного остовного дерева.

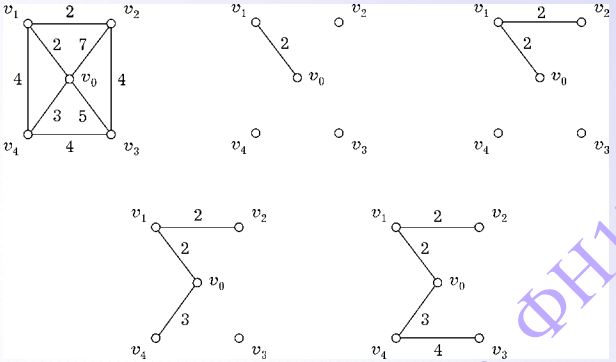

Рис. 10

Исходный граф изображен на рисунке *а*. На рисунке *б* проиллюстрирован результат выполнения первого шага алгоритма.

На *в* показан результат добавления следующего ребра $\{v_1,\ v_2\}$ с весом 2 из очереди. На *г* приведен результат добавления ребра $\{v_0,\ v_4\}$ с весом 3. Если следующим в очереди ребром будет $\{v_1,\ v_4\}$, оно будет отброшено.

Дальнейший ход работы алгоритма зависит от того, в каком порядке в очереди размещены ребра $\{v_2,\ v_3\}$ и $\{v_3,\ v_4\}$ с весами 4. Любое из них может быть добавлено в множество ребер остовного дерева, и на этом алгоритм закончит работу. На *д* приведено остовное дерево, полученное после добавления ребра $\{v_3,\ v_4\}$.

Для приведенного графа оба ребра с весом 2 войдут в остовное дерево независимо от порядка их расположения в очереди после сортировки, а ребро $\{v_1,\ v_4\}$ не войдет ни в какое остовное дерево наименьшего веса.

2.5. Топологическая сортировка.

Определение 2.15. Ориентированной сетью (или просто **сетью**) называют **бесконтурный ориентированный граф**

Сеть является бесконтурным графом, поэтому существуют вершины (узлы) сети с **нулевой полустепенью исхода** , которые называют **стоками** или **выходами сети**, и вершины (узлы) с **нулевой полустепенью захода**, которые называют **источниками** или **входами сети**.

Определение 2.16.
Уровень вершины сети — это натуральное число, определяемое следующим образом:

1) если полустепень захода вершины равна 0, то ее уровень равен 0 и наоборот (т.е. нулевой уровень N_0 — это множество всех входов);
2) если множества N_i вершин уровня i определены для всех $i \leq k$, то уровень N_{k+1} содержит те и только те вершины, **предшественники** которых принадлежат любому из уровней с номером от 0 до k, причем существует хотя бы один предшественник уровня k, т.е.

$$N_{k+1} = \left\{v\colon \Gamma^{-1}(v) \subseteq N_1 \cup \ldots \cup N_k,\ \Gamma^{-1}(v) \cap N_k \neq \varnothing\right\},$$

где $\Gamma^{-1}(v) = \{x\colon x \to v\}$ — множество предшественников вершины v.

Уровень вершины сети можно интерпретировать как **длину** максимального **пути** от входов сети до этой вершины.

Определение 2.17. Порядковой функцией сети $G = (V, E)$ называют отображение $\mathrm{ord}\colon V \to \mathbb{N}$, сопоставляющее каждой вершине сети номер ее уровня.

Во многих прикладных задачах возникает проблема такого упорядочения вершин сети, при котором вершины, принадлежащие одному уровню, располагаются друг под другом, а дуги ориентированного графа ведут в его изображении на плоскости от вершин с меньшим уровнем к вершинам с большим уровням слева направо.

Подобного рода проблема называется проблемой **топологической сортировки**, поскольку необходимо расположить вершины графа на плоскости так, чтобы отчетливо было видно распределение вершин по уровням.

Само расположение при этом может быть разным, лишь бы оно имело „слоистую“ структуру, в которой каждый слой составляют вершины одного уровня.

Топологическая сортировка применяется , например, при распараллеливании алгоритмов, когда по некоторому описанию алгоритма нужно составить граф зависимостей его операций и, отсортировав его топологически, определить, какие из операций являются независимыми и могут выполняться параллельно (одновременно).

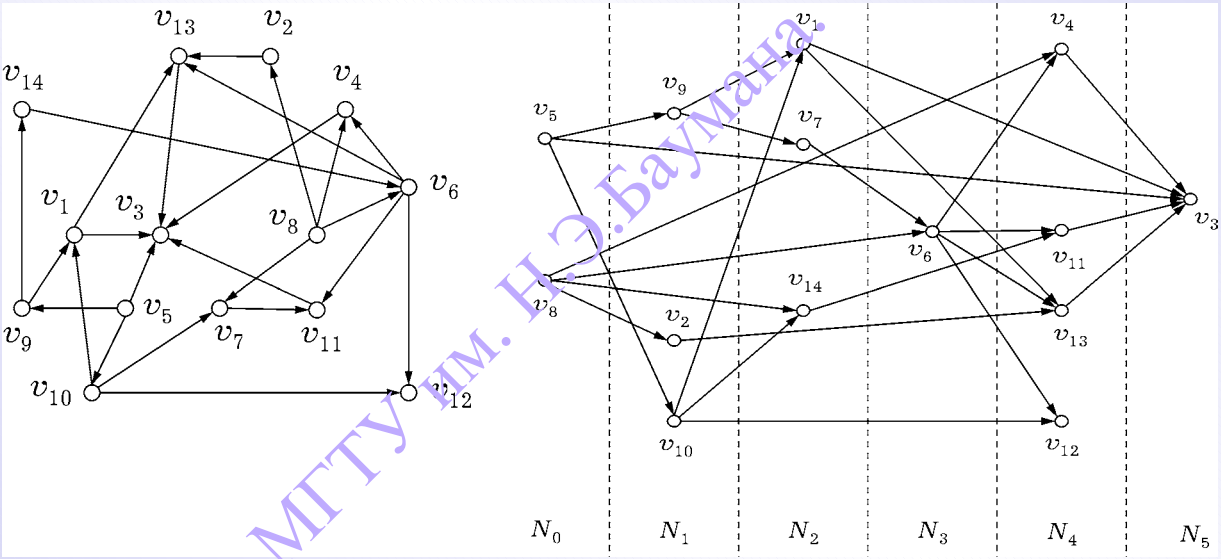

Рис. 11

Сеть и результат применения топологической сортировки сети.

Формально топологическую сортировку можно реализовать по-разному. Мы опишем один из возможных методов

Этот метод состоит в вычислении порядковой функции сети и известен как **алгоритм Демукрона**.

Предполагается, что вершины сети пронумерованы от 1 до n.

Наглядно процесс определения уровней вершин можно представить следующим образом.

Нулевой уровень образуют входы сети — вершины с полустепенью захода, равной 0.

Удалив из сети все вершины нулевого уровня и исходящие из них дуги, вновь получим сеть, входами которой будут вершины первого уровня исходной сети.

Указанный процесс „послойного“ удаления вершин следует продолжать до тех пор, пока все вершины исходной сети не будут распределены по уровням.

Алгоритм Демукрона использует матрицу смежности вершин B типа $n \times n$ в качестве средства представления сети и основан непосредственно на определении уровня вершины и свойствах матрицы B.

Сумма элементов k-го столбца матрицы B равна полустепени захода вершины v_k. Поэтому, просуммировав элементы матрицы по всем столбцам и выбрав вершины, соответствующие столбцам с нулевой суммой, получим множество вершин нулевого уровня — **входы** сети.

„Физически“ вершины и дуги из сети не удаляют, строки, соответствующие удаляемым вершинам,из матрицы B не вычеркивают.

Процесс вычисления порядковой функции.

Запишем **суммы элементов столбцов матрицы** B в **вектор** M длины n. Элемент m_k вектора M содержит полустепень захода вершины v_k.

Пусть из сети удалены вершина v_i и все исходящие из нее дуги.

Элемент b_{ik} равен 1, если из вершины v_i идет дуга в вершину v_k, и равен 0 в противном случае.

Чтобы пересчитать полустепени захода всех вершин сети, оставшихся в ней после удаления вершины v_i, надо из вектора M вычесть i-ю строку матрицы B.

Если на очередном шаге входами сети являются вершины $v_{i_1}, \ldots, v_{i_r}$, то для определения следующего „слоя“ вершин нужно из вектора M вычесть строки матрицы B с номерами $i_1, \ldots, i_r$ и зафиксировать**новые нулевые элементы** вектора M, появившиеся после вычитания. Фиксировать следует именно новые нулевые элементы, поскольку элементы вектора M, соответствующие вершинам, лежащим на предыдущих уровнях, стали равными 0 на предыдущих шагах алгоритма.

Порядковую функцию сети можно задать, указав множества вершин, принадлежащих каждому уровню, или сопоставив каждой вершине ее номер уровня. Первый способ более удобен при теоретических рассуждениях, второй — при вычислениях.

Алгоритм Демукрона вычисления порядковой функции сети

Алгоритм обрабатывает матрицу B смежности вершин графа порядка n. В результате работы алгоритма получаем массив Ord длины n, i-й элемент которого равен номеру уровня вершины v_i.

0. Сформировать множество V_1 вершин сети. Значение счетчика уровней k положить равным 0. Найти суммы элементов по всем столбцам матрицы B (полустепени захода вершин) и заполнить ими массив M.

1. Если множество V_1 не пусто, перейти на шаг **2**,если иначе, то перейти на шаг **5**.

2. Определить множество I номеров всех новых нулевых элементов массива M, т.е. таких, что соответствующие этим номерам вершины принадлежат множеству V_1.

3. Присвоить элементам массива Ord с номерами из множества I номер уровня k и удалить вершины с этими номерами из множества V_1 („замаскировать“ вершины). Вычесть из массива M строки матрицы B, соответствующие вершинам с номерами из множества I (т.е. вершинам последнего вычисленного уровня).

4. Увеличить счетчик уровней на 1 $(k := k + 1)$. Вернуться на шаг **1**.

5. Закончить работу.

Пример 2.5. Применим алгоритм Демукрона к сети, представленной на рисунке

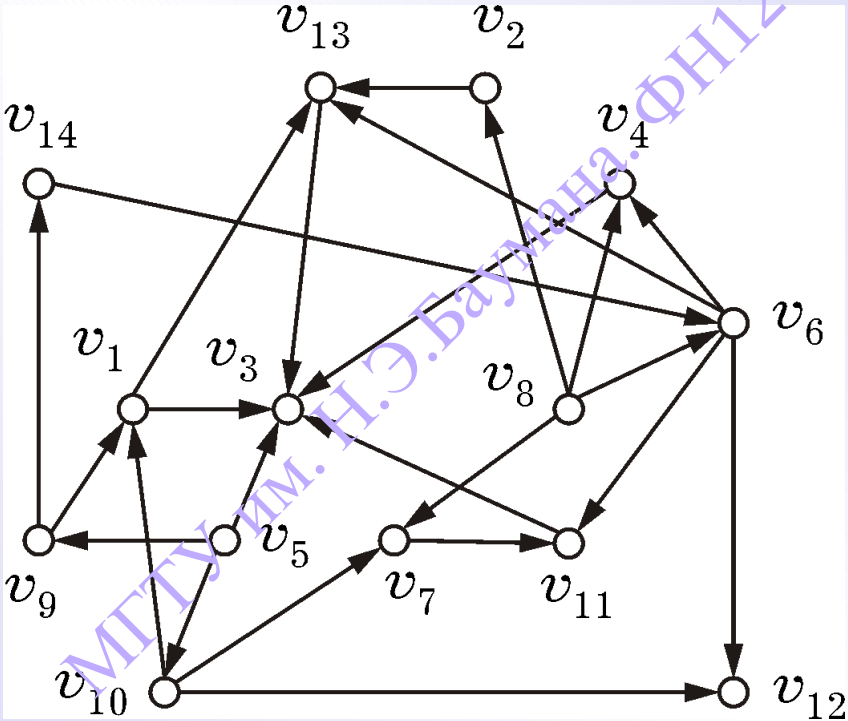

Рис. 12

Матрица смежности вершин сети имеет следующий вид:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14
1			1										1	
2													1	
3														
4			1											
5			1						1	1				
6				1							1	1	1	
7											1			
8		1		1		1	1							
9	1													1
10	1						1					1		
11			1											
12														
13			1											
14						1								

Покажем последовательность значений массива M, соответствующую итерациям алгоритма и множества N_i вершин i-го уровня. Прочерки соответствуют вершинам, не принадлежащим множеству V_1 („замаскированные“ вершины) на соответствующем этапе алгоритма.

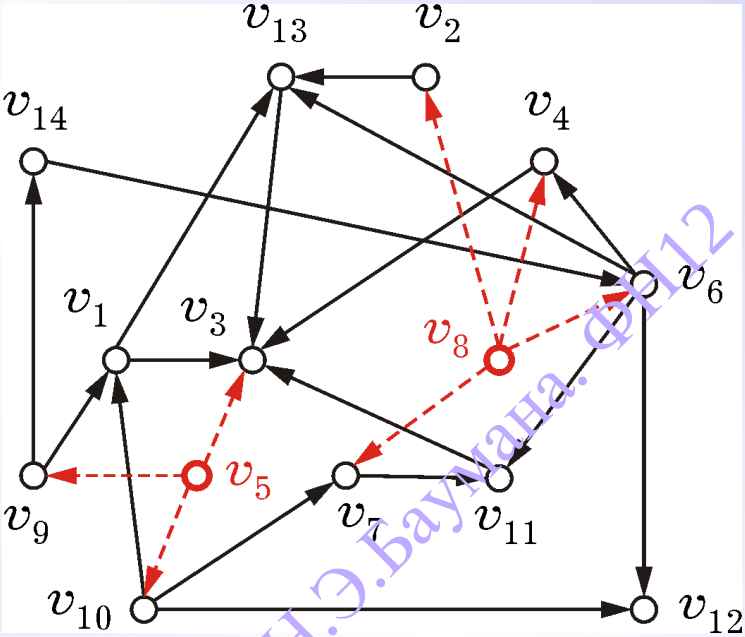

Рис. 13

1	2	3	4	5	6	7	8	9	10	11	12	13	14	
2	1	5	2	0	2	2	0	1	1	2	2	3	1	$N_0 = \{5, 8\}$

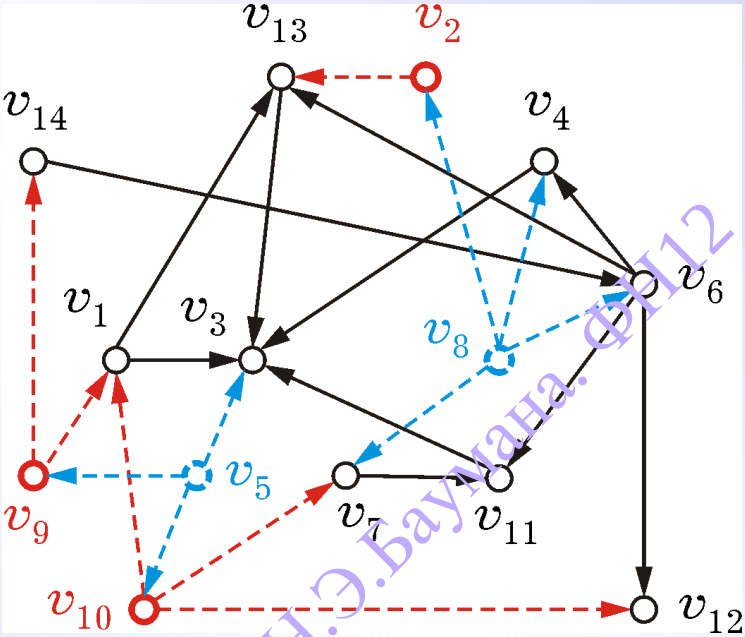

Рис. 14

1	2	3	4	5	6	7	8	9	10	11	12	13	14	
2	1	5	2	0	2	2	0	1	1	2	2	3	1	$N_0 = \{5, 8\}$
2	0	4	1	–	1	1	–	0	0	2	2	3	1	$N_1 = \{2, 9, 10\}$

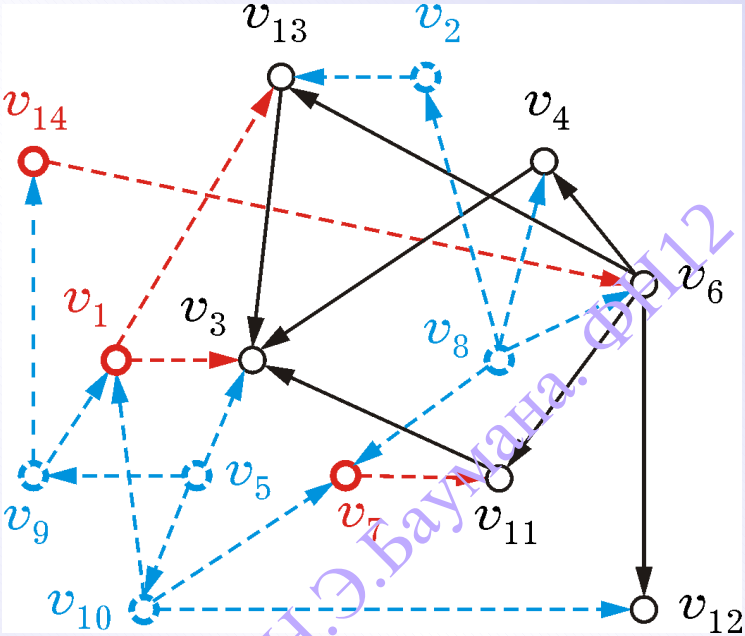

Рис. 15

1	2	3	4	5	6	7	8	9	10	11	12	13	14	
2	1	5	2	0	2	2	0	1	1	2	2	3	1	$N_0 = \{5, 8\}$
2	0	4	1	–	1	1	–	0	0	2	2	3	1	$N_1 = \{2, 9, 10\}$
0	–	4	1	–	1	0	–	–	–	2	1	2	0	$N_2 = \{1, 7, 14\}$

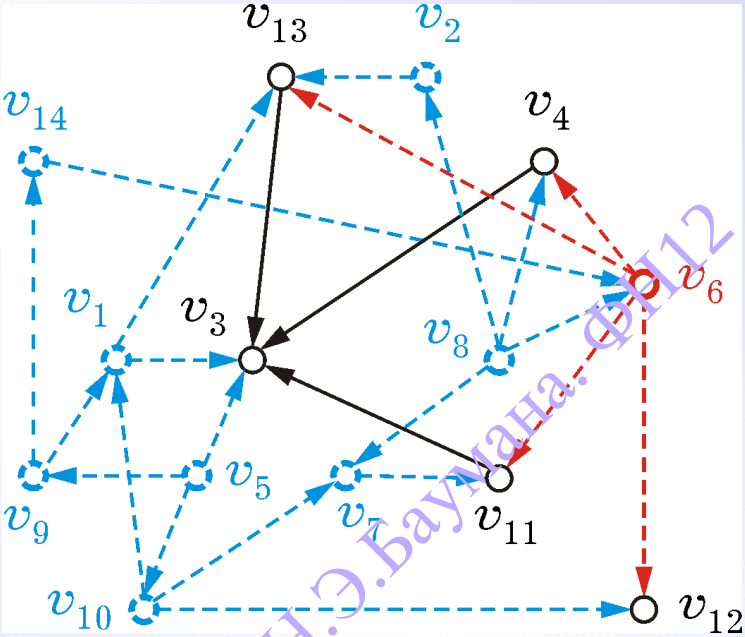

Рис. 16

1	2	3	4	5	6	7	8	9	10	11	12	13	14	
2	1	5	2	0	2	2	0	1	1	2	2	3	1	$N_0 = \{5, 8\}$
2	0	4	1	–	1	1	–	0	0	2	2	3	1	$N_1 = \{2, 9, 10\}$
0	–	4	1	–	1	0	–	–	–	2	1	2	0	$N_2 = \{1, 7, 14\}$
–	–	3	1	–	0	–	–	–	–	1	1	1	–	$N_3 = \{6\}$

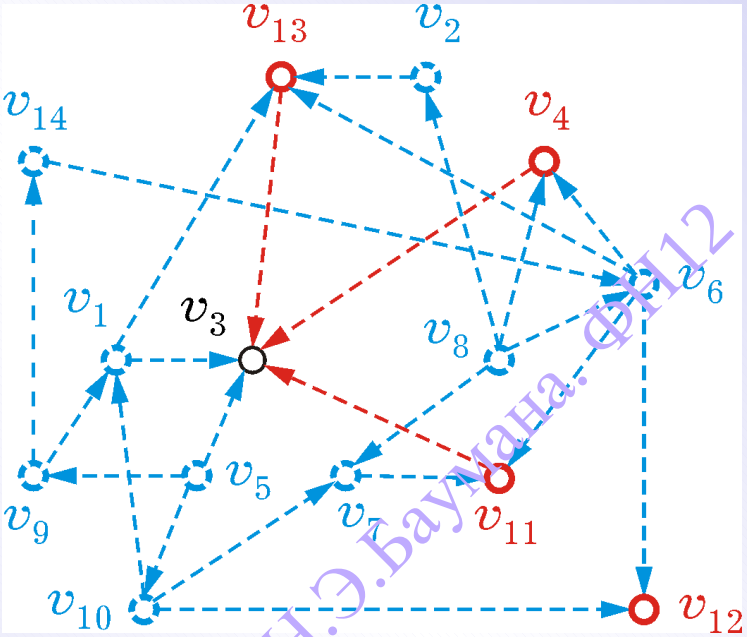

Рис. 17

1	2	3	4	5	6	7	8	9	10	11	12	13	14	
2	1	5	2	0	2	2	0	1	1	2	2	3	1	$N_0 = \{5, 8\}$
2	0	4	1	–	1	1	–	0	0	2	2	3	1	$N_1 = \{2, 9, 10\}$
0	–	4	1	–	1	0	–	–	–	2	1	2	0	$N_2 = \{1, 7, 14\}$
–	–	3	1	–	0	–	–	–	–	1	1	1	–	$N_3 = \{6\}$
–	–	3	0	–	–	–	–	–	–	0	0	0	–	$N_4 = \{4, 11, 12, 13\}$

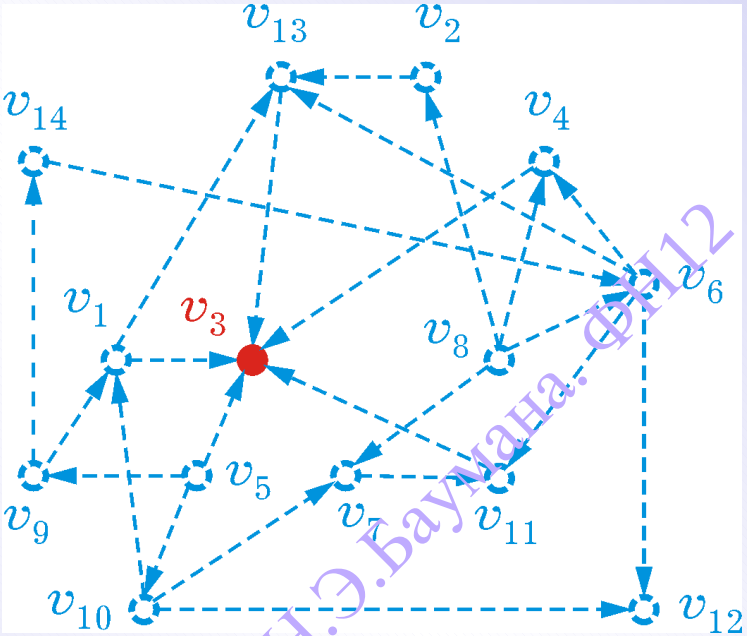

Рис. 18

1	2	3	4	5	6	7	8	9	10	11	12	13	14	
2	1	5	2	0	2	2	0	1	1	2	2	3	1	$N_0 = \{5, 8\}$
2	0	4	1	–	1	1	–	0	0	2	2	3	1	$N_1 = \{2, 9, 10\}$
0	–	4	1	–	1	0	–	–	–	2	1	2	0	$N_2 = \{1, 7, 14\}$
–	–	3	1	–	0	–	–	–	–	1	1	1	–	$N_3 = \{6\}$
–	–	3	0	–	–	–	–	–	–	0	0	0	–	$N_4 = \{4, 11, 12, 13\}$
–	–	0	–	–	–	–	–	–	–	–	–	–	–	$N_5 = \{3\}$

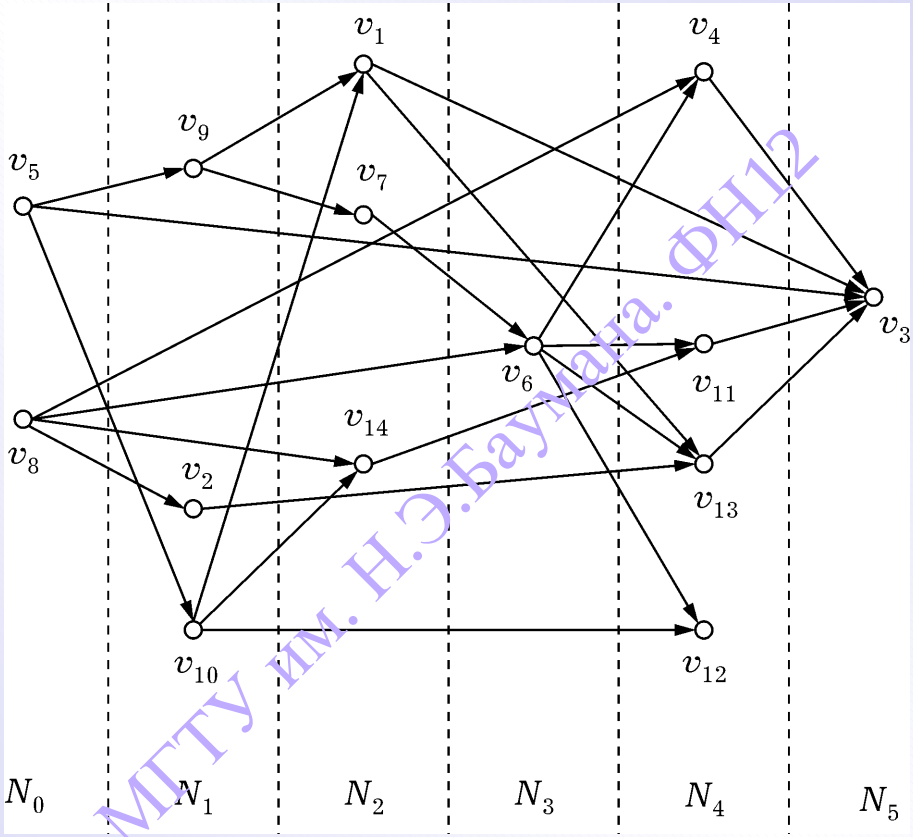

Рис. 19

Результат применения топологической сортировки сети.

Алгоритм Демукрона может быть модифицирован так, чтобы он останавливался, если ориентированный граф, поданный на вход, не является сетью, и сообщал об этом. Можно увидеть, что анализируемый граф не будет сетью тогда и только тогда, когда при очередном перевычислении массива M не появятся новые нули.